

**METHOD, SYSTEM, APPARATUS AND
CONTENT MODEL FOR THE CREATION,
MANAGEMENT, STORAGE, AND PRESENTATION
OF DYNAMIC OBJECTS**

5

Priority

This application claims the benefit of U.S. Provisional Application No. 60/228,853 filed August 29, 2000, which provisional application is hereby incorporated by reference in its entirety.

10

Field of the Invention

The invention relates to methods, systems, apparatus and content models for the creation, management, storage, and presentation of dynamic objects, as exemplified by, but not limited to streaming audio and streaming video.

15

Background of the Invention

There is a continuing need to facilitate collaboration and interaction among individuals who may be widely separated by distance, time, or both. A 'rich media' format, delivered instantaneously via the Internet or corporate intranets, enables organizations of all sizes to communicate quickly and effectively with customers, partners and employees. Fast, effective communications drives top-line revenue, streamlines business processes, and lowers cost.

20

The method, system, and apparatus of the present invention provides the ability to organize, enrich, and leverage existing and future rich media content, including streaming content.

25

Summary of the Invention

The invention relates to methods, systems, apparatus and content models for the creation, management, storage, and presentation of dynamic objects, as exemplified by, but not limited to streaming audio and streaming video. The methods, systems, and

30

apparatus are built upon a content model for rich media formats, that is, media formats adapted to dynamic, time changing objects or time based content models.

The content model provides a hierarchy of views for the same content. Thus, one view might be an “in time” view where objects are presented in time order, while another
5 view may be an “out-of-time” with the objects shown as for editing, the content model supporting a number of different view structures.

The content model provides a hierarchy of objects, with a container object at the top, as a “Talk Folder” object, and a plurality of time-based objects, called element
10 objects, in the container object. The individual element objects have properties, in the OOP sense, such as start time, end time, and linked documents. The linked documents are logical functions, such as audio-visual objects or elements, transcript objects or elements, and graphics objects or elements. These objects or elements, themselves, have associated detailed characteristics such as a construction or build, type data, bandwidth detail,
15 JavaScript details, and track objects. The next levels down in the hierarchy are those things that will actually play. The track object builds in a time view, parses the objects, and builds the page.

Brief Description of the Figures

20 FIGURE 1 is a screen shot of an in-time view of a talk-play screen.

FIGURE 2 is a screen shot of an out of time view of a talk view screen.

FIGURE 3 is a screen shot of the edit slide screen.

25

FIGURE 4 is a screen shot of the “New/Edit Screen” function.

FIGURE 5 is a screen shot of the “Single Choice Vote Screen”.

30 FIGURE 6 is a screen shot of the “Multiple Choice Vote Screen”.

FIGURE 7 is a screen shot of the “Poll Results Screen, Single Choice”.

FIGURE 8 is a screen shot of the “Poll Results Screen – Multiple Choice.”

5 FIGURE 9 is a screen shot of the “Create Shortcut” screen.

FIGURE 10 is a screen shot of the “Select Shortcut Target” screen.

FIGURE 11 is a screen shot of the “Browse” view for shortcut object.

10

FIGURE 12 is a diagram showing the relationship between the “Talk Folder” container object, the element objects, the related objects, and the track objects.

15

FIGURE 13 is a diagram illustrating one type of system, including database servers, application servers, web servers, internets and intranets, and browsers.

FIGURE 14 illustrates a request life cycle according to one embodiment of the invention.

20

FIGURE 15 illustrates an Object architecture according to one embodiment of the invention.

Detailed Description of the Invention

The method, system, and apparatus of our invention provides total management, storage, searching, calling, and delivering of time dependent rich media content.

25

Specifically, the invention utilizes time dependent objects and time dependent object management systems using a defined object model, and utilizing rich media, that are capable of real time interactions. One exemplary form of rich media is streaming media. A further aspect of the invention is the Management of Time Dependent Objects, with the method, system, and apparatus of the invention assembling time based objects on the fly, in real time, for delivery to a browser or client. Exemplary is a rich media player receiving and presenting pages on the fly from media objects.

30

5 The method, system, and apparatus of the invention includes a scalable software platform that manages, delivers and enriches all forms of rich media and streaming content for various end user applications. It accomplishes this by converging streaming media, documents, graphics, Web links and other communications content into a personalized, interactive and extensible communications format.

The server, data structure, and object model of the method, system, and apparatus described herein provide:

10 Central management of all streaming and rich media content. This is accomplished through central organization, management and delivery of all streaming and rich media content, including Eloquent, Microsoft Windows Media, Real Networks, and QuickTime formats.

15 Built-in profiling and personalization technology that enables users to push relevant information to individuals and groups as well as to request that certain types of information be automatically delivered to them. Library and category views offer users more structured views of all the content. Comprehensive search functionality enables users to find content ad-hoc, either through a powerful full-text search capability or alternatively by specific metadata types, such as author, organization and time.

20 The method, system, and apparatus described herein provide a User Interface that is both pleasing and functional within a standard Web-based (browser) paradigm. The four access points into the method, system, and apparatus of the invention are: My View, Search, Browse and Library.

The My View section includes the following categories:

- a. Favorites - frequently used files or folders, personally selected by the individual user and displayed in a library-type view.
- 25 b. Subscriptions - content types selected by the user to be automatically be added or updated.
- c. Recommendations - content delivered to the user by someone else in the system based on his/her profile.

The Search section enables users to perform a full-text search of content or to search for content based on metadata such as author, date, time, etc.

The Browse section provides categories by subject matter categories.

5 The Library section presents all the content in the system organized in a typical folder hierarchy.

The data structure, the metadata, and the set of graphical user interfaces facilitate end-user interactivity and provide an extensible framework for working with rich media, including streaming media. In this way, rich format content, including streaming media, go from being the end-solution, to becoming the starting point for collaborative, ongoing
10 communications that grows in value as more users view the content and interact with it. This new framework enables users to add threaded discussions, documents, URLs, polls, and other communications content to existing streaming and rich media, enhancing the overall communication.

Attached Items and Discussion Topics: One embodiment of our invention provides a
15 graphical user interface panel called: *Attached Items* and *Discussion Topics*. Documents, polls, shortcuts and links to other content (URLs) can be added and synchronized in the *Attached Items* section, while topics of interest about the content can be added in the *Discussion Topic* area. The user decides when the attached content should appear - and disappear. The user can also decide whether it is mandatory that the viewer read a
20 document or participate in a discussion or Poll before continuing through the presentation.

Polls: Polls can be used to survey a site's viewing audience. Anyone with appropriate permissions can set up a poll and link it to a specific part of a presentation. Results can be made public or restricted, depending on the permissions set by the author.

25 Shortcuts: A shortcut is a pointer or reference to another item in the system. Shortcuts are useful to reference objects and items that physically exist in another place on the network.

A further aspect of the method, system, and apparatus of our invention is enterprise-level security and access control. The system, method, and apparatus centrally manage

enterprise-wide access. Access controls include single user login to access any information that authorized users can view, alter or create content, and secure extranet access for remote users, suppliers and customers.

A still further aspect of the invention is enterprise scalability.

5 The method, system, and apparatus of our invention manages and delivers all forms of streaming and rich media content, including Eloquent, Microsoft Windows Media, Real Networks, and QuickTime formats.

10 In a preferred embodiment of our invention, the system, method, and apparatus are all completely standards-based and require only a browser on the desktop. To be understood, however, is that all streaming formats - including Microsoft Windows Media and Real Networks - require that a 'player' be installed on the client side in order to deliver that content.

15 The rich media and rich media management features of the method, system, and apparatus provide dynamic and collaborative communications. Communications are dynamic. Every good presenter knows this. In addition, communications in business is inherently collaborative. The audience adds to the value of the communication with questions, updates and ideas related to the topic. The method, system, and apparatus of our invention not only effectively deliver content to large audiences; it also facilitates and captures the interactive information sharing that naturally occurs in conjunction with
20 those presentations.

By simply communicating, users create content that adds to the richness of the experience. Streaming media becomes rich. And rich media becomes richer. This interaction builds collective intelligence and increases the value of the content over time.

25 The data structure and object structure of the method, system, and apparatus of our invention begins with a "folder." Folders are logical 'containers' that are used to organize communications and content according to any criteria that the user desires. Information in a folder can be organized by topic, by date, by intended audience, or simply at random. Users can set permissions on folders that by default will pertain to the content contained

in them unless the individual items are given different permissions. Folders appear in the Library section.

5 The objects contained in the container "Folder" can be viewed in various ways or views. One such view is the "overview" The overview function allows users to quickly see how time-based content such as streaming media is organized *over time*. The overview presents thumbnails of key 'scenes' of the streaming presentation - defined by the slides in the presentation for most business presentations. It then shows which additional communications content (transcript, documents, URLs, polls, etc.) is associated with those scenes. It also lists the start and end times for all elements of the communication.

The overview function enables content owners to easily review, edit and maintain a piece of rich media content. It gives users a convenient way to find and edit attachments and permissions associated with a presentation.

15 Another aspect of the rich format media of the invention is a threaded discussion. A threaded discussion is a shared text file that enables multiple users to post text-based comments in a common viewed format. Individual users can follow a 'discussion' among group members over time, can comment on individual parts of the conversation, and can begin new 'side conversations' or threads.

20 Unlike email, where a user might get many similar responses on the same topic because no one knows who else has responded to the original mail, a threaded discussion enables participants to see who else has added to the discussion -- and what they said -- in real time. This enables a widely dispersed group to have an electronic conversation that is both more natural and more useful than email.

25 Scalability is provided through a wide range of configuration options designed to fit any corporate or other user requirement. The system's Java Application Server uses clustering technology to grow capacity as large as the user base requires. Databases can be run locally or can be configured to run remotely. Remote databases can include supported enterprise databases running on the UNIX operating system. Finally each media server (Eloquent, Microsoft Windows Media, Real Networks, and QuickTime) can

be operated independently and scaled to support the requirements for streaming media capacity.

The method, system, and apparatus of the invention are readily customizable, for example, through the use of customizations and third-party applications. This is the result of a completely object based structure, which facilitates adding new object types to the system and automatically benefiting fully from the existing server infrastructure, including permissions, categorization, subscriptions and full text indexing. The system also supports custom indexing and the addition of new actions that function in a fully integrated fashion.

Customizability and scalability also arises from object implementation. Security permissions and constraints can be inherited from a defined group. Security is implemented independently for each content object in the system, so that a single rich media presentation can present a different set of supporting information and links to different audiences - and can do so easily and transparently.

The method, system, and apparatus of the invention provide and facilitate a full suite of services, including rich media content production and complete communications design and implementation services on a per project basis.

This results in a more collaborative environment around talks, along with a framework for adding value to talks over time, for example by allowing association of arbitrary objects with intervals within Talks. All object types within the method, system, and apparatus of the invention have the capability of being 'time-sensitive'. This allows the association of a start time and end time with any object instance. When contained within an object that is time-sensitive, like a Talk (or Talk Folder), the container can use the time attributes of the objects to determine when it is appropriate to display them.

Talks are central to the method, system, and apparatus of our invention. Talks are container objects or 'Folders' within the structure of the method, apparatus, and system of the invention. This allows objects to be created inside of talks.

The method, system, and apparatus of the invention are built upon the object model, shown in FIGURE 12. The principal organization element in the method, system, and apparatus of our invention is the "Talk." The "Talk" is an object of the container

class. FIGURE 12 illustrates a “Talk” container class object, 101, containing three element objects. The element objects provide certain properties, inherited by the next level of objects. These properties include start time, stop time, and duration. These time dependent properties are inherited by content objects, such as the AV Object or audio-visual object, 121, the Transcript Element Object, 123, and the Graphics Element Object, 125, which contain Javascript files, time synchronized text, and time synchronized graphics. The content objects are controlled or tracked by tracking objects, 132, 133, and 135, which provide control data, build information, type information, and bandwidth information.

10 Exemplary objects include:

Polls
Documents
Shortcuts
URLs, and
15 Folders.

Any object created within a Talk ‘Folder’ will contain fields for entering start and stop time offset for that object within the talk. This is illustrated in FIGURE 12. These fields will appear on the normal edit pages for these objects. Moreover, everything within a talk folder is considered part of the talk. Talks folders implicitly include all of the ‘normal’ items. One discussion can be associated with each talk.

Content administrators can enable the discussion by selecting the ‘Enable Discussion’ action on the talk. The discussion will be an implicit member of the Talk; it won’t be explicitly displayed as a separate object.

A talk folder contains at least two views, an “out-of-time” view and an “in-time” view. FIGURE 1 is a screen shot of an in-time view of a talk-play screen. The in-time view is a player. The out-of-time view is a vertical timeline expansion of all the presentation objects. FIGURE 2 is a screen shot of an out of time view of a talk view screen.

Since the talk folder is a time-sensitive container, the new/edit screen has fields for the start/stop time of the object being created. Default time start/end is the duration of the current slide, but this is editable. Once an item is added to a talk folder, a Control

frame updates after the item is added, and displays a list of discussion topics associated with the current time.

The Control screen also displays a 'Add a new topic' link that invokes the 'create topic' screen. The "create topic" screen will act just like a selection from the 'new' menu, and a list of the objects associated with the current time, with a link to their open verb will appear.

Controls frame changes as the talk progresses, displaying the information that is appropriate to that point in the presentation. The window is implemented using DHTML, so all of the content to be displayed will be downloaded in advance. This allows refreshing the information without a server hit.

The talk/view screen or window allows a user to get an overall view of the talk and the objects associated with it. It is useful for a content administrator to bootstrap the conversation, displays all the slides in the talk and associated times, and displays attached objects within the appropriate slide with separate timing info.

If the duration of the object overlaps the slide time at all, it is shown in with that slide. This means that a single object could be displayed for part of a slide, an entire slide, more than one slide, or even for the entire duration of the Talk.

Clicking the start or stop time for the object brings up the edit screen to allow editing of the start/stop time values, as well as those fields normally accessible on the edit screen for that kind of object. Clicking the object name 'opens' the object. Clicking the slide key jumps to that point in the talk (in-time view) or shows a full-size image of the slide, thereby causing creation/attachment of new objects, and shows transcript snippets for each section.

Clicking the 'Edit Slide Data' will bring up the Edit Slide Screen.

The "Edit Slide Screen" is shown in FIGURE 3. The "Edit Slide Screen" opens a window that allows actions to be executed on the items attached to a slide in a talk. This is required to edit the objects or manipulate permissions. A pop-up of all the slides in the talk is displayed. Slide titles are displayed in the popup in the order they appear in the talk. Selecting a slide and hitting the 'Go' button will allow the user to edit that slide.

The window displays the current slide. The start time and end time for the slide are displayed below the slide.

Clicking on other keys allows loading the talk (frameset) at that slide, or displays a full-size version of the slide.

Screen shots for the Poll Object are shown in FIGURE 4 for the “New/Edit Screen” function, FIGURE 5 for the “Single Choice Vote Screen”, FIGURE 6 for the “Multiple Choice Vote Screen”, FIGURE 7 for the “Poll Results Screen, Single Choice”, and FIGURE 8 for the “Poll Results Screen – Multiple Choice.”

The Poll Object allows surveying the users and displaying the information in a chart. Poll objects are ‘first-class’ objects like documents and URLs, and can be created anywhere in the system. Just like all other objects, they have permissions, and can be used in recommendations, etc.

The Creator can configure if users can change their votes. When the poll object is called, the system records each users vote. Users are not allowed to vote more than once, although they may be able to change their votes. The creator can configure if and when users can see results, such as Always; (Users can see results before and after they vote), Never; (Users cannot see the results) After, (Users can only see results after they’ve voted), and After poll ends; (Users can only see results after the poll has ended). The Creator can always see results. The Poll object also allows the creator to suspend the poll. This allows no one new to vote.

In running the poll object, the creator must specify a question and at least two responses, and whether the users must pick exactly one or ‘zero to n’ responses. The form implementation may be radio buttons or checkboxes depending on which option is selected.

In the poll object, Read permission is required to see the results (where available), and write permission is required to vote, while Edit permission is required to change options on the poll.

Vote screen (single-choice) is shown in FIGURE 5. This is the view that a user with write access who has not voted would see. Vote’ submits the users vote. If the user has already voted, clicking the object will show the poll results if the user is allowed to see them. Otherwise it will show an informative message indicating that the user has already voted.

If the 'allow user to change vote' flag is enabled, the button reads 'Change Vote' instead of just 'Vote'. The percentages shown by each item are the number of respondents who voted for that item. They should add up to 100% (+/- fuzz).

5 The Multiple Choice Vote Screen, shown in FIGURE 6, looks and works identically to the previous screen except that the choices are represented by checkboxes rather than radio buttons. This allows the user to select more than one item. The percentages shown by each item are the number of respondents who voted for that item. They do NOT necessarily add up to 100%.

10 The Poll Results screens, FIGURE 7 for the Single Choice and FIGURE 8 for the Multiple Choice, are displayed if the user has permissions to see the results and the Poll is configured to allow that. Percentages work as previous described. Users with control permission to the Poll (Admin, creator, etc) can always see the results.

15 The Poll Results Multiple Choice screen, shown in FIGURE 8, is displayed if the user has permissions to see the results and the Poll is configured to allow that. Percentages work as previous described. Users with control permission to the Poll (Admin, creator, etc) can always see the results.

20 The Shortcut object is a pointer or reference to another object in the method, system, and apparatus of the invention. Shortcuts are very useful in cases where you want to reference objects in one place, but don't want to move or copy the object. They are very similar to Windows shortcuts. When displayed in the browse screens, shortcuts use the icon of the object they are pointing at.

FIGURE 9 shows a screen shot of the "Create Shortcut" screen, while FIGURE 10 shows a screen shot of the Select Shortcut Target Screen.

25 Shortcuts, as first-class objects, can have permissions. The permissions on a shortcut only control access to the Shortcut, not to the target object. In order for a user to click on a shortcut and open the target object, they must have read permission to both the shortcut and the target object. If a shortcut is created that a user does not have permission to see, that does not necessarily preclude them from opening the target object directly.

When shortcut objects are full-text indexed, only the metadata for the object is indexed, not the content of the target object.

The Create Shortcut screen, FIGURE 9, is a normal edit screen with name and description. A non-typeable box contains the name of the target object. The shortcut can
5 be directed to a new object by clicking the 'Select Object to Shortcut' link which invokes the 'Select Shortcut Target' screen in a new window. If no name is input, the name defaults to 'Shortcut of ' + the name of the target object. The target object is the only required field

The Select Shortcut Target screen, shown in FIGURE 10, allows the user to
10 browse and select any object in the system.

Selecting an object closes the window and fills in the appropriate fields in the Create or Edit Shortcut screen. Hitting the cancel link closes the window.

Inside the browse view (or any other view), shown in FIGURE 11, the shortcut object looks, acts, and behaves like its target object. One exception is that the actions
15 that appear on the Shortcut modify the Shortcut rather than the target object. Clicking on the shortcut will behave exactly as clicking on the original object with one exception- two audit records will be generated; one for opening of the shortcut and one for the original object.

Key to the method, system, and apparatus of our invention is the object structure.
20 In order to have a standards-based, extensible, indexable and searchable persistent object the objects and data structures are architected as follows:

The Objects are serialized to XML. This provides a standards based storage format. That is readable, editable and easily parsed. The serialized Objects are stored with an SQL database.

25 The method, system, and apparatus of the invention need not use conventional SQL Tables with columns matching the object fields and indexes to access the objects. Instead, to facilitate extensibility we store the object definition as a blob of XML in a single static table. To get indexability of this static table we use a separate indexing table that is capable of indexing an unlimited number of different fields all at once.

30 Traditionally we would have to add more columns to a growing number of SQL tables

and growing number of indexes on these tables to store and index all of this type of data. Using our scenario we never change tables to add columns or add indexes to index more fields.

5 Indexability .. As the number of fields grows we would have to be continually adding indexes to locate and retrieve objects from the persistent store. Each object defines what fields it needs to have indexed. These fields are indexed with a single SQL table.

10 Searchability .. Objects contain searchable content as well as indexable fields so the Object System allows objects to have their content full-text indexed as well. Each object specified what content is to be indexed along with the Object.

One embodiment of the apparatus and system of the invention are illustrated in FIGURE 13, which shows a web-based exemplification. The system and apparatus, 201, has a web browser, receiving data, as pages and data, over an internet 213 or intranet, from a web server, 215. The web server, 215, receives data from an application server, 15 221, which, in turn, receives data from a source of streaming media, 223, and a database server. The database server, 231, interacts with various databases, 241 and 243, as well as tools repositories, 251, which may contain tools, applets, servlets, business objects, and the like.

The method, system and apparatus of our invention are also built on standards. 20 These include Java JDK 1.2.2 (Java2), Java Servlets 2.2, Java Server Pages 1.1 (JSP) JDBC 2.2, Extensible Markup Language (XML), Java Messaging Service (JMS), Java Naming Service 1.2 (JNDI). The method, system, and apparatus of our invention is also built upon and synergistically integrates third party tools, such as WebLogic Server 5.1.0 sp4+, Microsoft SQL Server 7.0 sp1+, Oracle 8i (8.1.6), Sybase Adaptive Server 6.0, 25 Verity 4.2, and Xerces (Apache) XML Parser 1.1.1.

The servlet is named /ecs by default, and is a singleton that lasts for the time the WebLogic server is running. A request in the system, method, and apparatus of our invention completely manages all HTTP GET and POST requests. Requests must specify at least an Object and a method, such as /ecs?ID=<objectID>.<method> or 30 /ecs/<objectID>/<method>. The servlet lasts for the duration of a request.

FIGURE 14 illustrates a request life cycle according to one embodiment of the invention.

Methods in the method, system, and apparatus of our invention control all interactions with objects. That is, all interactions with objects are through methods.

5 Methods act on objects, and are also known as Verbs or Actions. A method lasts for the Duration of a Request. The base class of methods is Emethod.

Everything in the method, system, and apparatus of our invention is an Object. An object is the basic data entity, and it encapsulates all related logic. Objects are persistent. The base class is the EObject

10 FIGURE 15 illustrates an Object architecture according to one embodiment of the invention.

The Object architecture includes object stores, object caches, an object indexer, an object monitor, an XML Parser, and an XML Writer. Further aspects of the Object architecture include a full-text indexer, cache synchronization, a task manager, auditing
15 functionality, reference management, and a document store

The Method architecture of the method, system, and apparatus of our invention requires that all GUI Initiated Interactions be managed by Methods, that methods perform actions on objects, and that methods produce a GUI output. Further aspects of the Method Architecture include performing input checking, parameters, enabled status, permissions,
20 privileges, object state, executing, gathering GUI related data, and performing the specified action on the specified object(s). Still further aspects of the method architecture including specifying the GUI output, JSP File, issuing Errors, Warnings or Info, operating on content, and redirecting.

The Object Store stores all persistent objects, is RDBMS (SQL) Based, and
25 operates on XML (Writer and Parse) and the SQL Table:

The Object Cache functions by lookup by Object Ids, controls object lifetime management, and implements cache synchronization when using WebLogic Clusters

The Object Indexer manages the Attribute Indexer, the Attribute Search Engine, the Attribute Query, the Object Indexer, the Attribute Indexer, the Attribute Search
30 Engine, the Attribute Query 'Language, such that object implementations specify indexed attribute fields, and the SQL Table:

Object Monitors are registered and monitor the creation all new objects, and the updates to existing objects.

The Full-Text Indexer is a content indexer, content search engine, and a content query language. Object implementations specify attribute fields and content to be
5 indexed.

The Document Store stores all managed content. It is file system based, and may be full-text indexed.

The Auditing function is such that methods can audit the actions they perform on objects and on themselves, as well as system actions.

10 Reference Management handles deletion of objects, as well as the registration of how objects play in the system, and what other objects the objects own or contain.

A WebLogic Property File files weblogic.properties, controls the WebLogic configuration and required changes, registers servlets, and is JSP servlet enables. It also controls cluster configuration, JNDI Configuration, and JMS Configuration.
15

The Property Files
file: system and method .properties, file: system and method .overrides.properties, any module related property files, control all system and method related configuration, control registered objects, control registered methods,
20

Implemented Object Types include Library, Folder, URL, Shortcut, Category, Search, Category, Poll, Discussion, Topics, Replies, Read List, Users, Groups, RMS Repository, File System Repository, Microsoft Media Server Repository, and Real Media Server Repository.

25 The Permissions Levels on Objects include, Read (Open, View), Write (Populate, Participate), Edit (Modify Attributes), and Delete (Control Permissions).

In the method, system, and apparatus of our invention, access control lists are provided. The Access Control Lists are attached to each Object, and contains entries for users and groups who have some type of access to the object. Each entry specifies a
30 permission level (RWED). In the method, system, and apparatus of our invention,

Permission ACLs are subject to the following rules: granted permissions are

additive, inherited from containing object, can be 'indirected' to another object,

Closely related to permission Access Clearance Lists are Extended Object Permissions, including Hide Before (Effective Date), Hide After (Expiration Date), and Hide Always.

5 The method, system, and apparatus of the invention provide Privileges on users. These include Bypass, Admin, Login, and Alter.

 User Password Management modules specify minimum number of letters, minimum number of digits, minimum and maximum lengths, history lists, expiration dates, pre-expired passwords, and stored one-way encrypted.

10 System Management modules within the method, system, and apparatus of our invention include utilities for database initialization, utilities for database upgrade, MD5, GUID – Unique ID Management, RMS Usage Reporting, Database Import, and CSV User Import.

 User customization includes capability for and support of, page decoration
15 overrides, company logo (GIF, URL, alt), top-level labels (i18n), HTML includes, company logo area (below company logo), login page banner (above fields), tab configuration, which top-level tabs, my view, search, browse, library, and order of top-level tabs, among others.

 Other utilities support authentication, including creating users and groups, user
20 logs, public access and guest accounts, and sign up, a REMOTE_USER Header for Login User Name withNo Password Required, Automatic User Creation when REMOTE_USER is detected.

 The system servlets are singletons and include requests, session/login management, access target object, method dispatcher, render JSP, return content, render
25 errors, redirect, method, check parameters, check enabled, check permissions, gather data, perform actions, specify jsp*, and request life cycle.

 Objects include, cache, manager, object, store, attribute, indexer, cluster, cache, manager, writer/parser, object, monitor, full-text, document, store, verity, attribute, indexer, reference, manager, file system, document, store, audit, object, store, registered,
30 and monitors.

Database Schema

The system, method, and apparatus of the invention rely on a relational database to store Object attribute information as well as audit and tracking data. Oracle version 8 and MS-SQL Server 7 are both supported. One of the more significant differences between Oracle and SQL7 is in their support of identity columns (columns which are to be assigned numerically increasing unique values). SQL7 supports the “identity” keyword, whereas Oracle requires the use of a sequence. Since this affects how code is written, inserts into tables with an identity column are performed through stored procedures, allowing our Java code to be the same regardless of the database system being used. Other database-specific constructs are isolated to views (for example, EUsageSummaryView), again relieving the Java code of having any database-specific knowledge.

| SQL7/Oracle datatype mapping | |
|------------------------------|--|
| SQL7 | Oracle |
| varchar | varchar2 |
| int | integer (translated by Oracle to number(38)) |
| datetime | date |
| text | long |
| image | blob |

Tables

Legend:

PK – Primary Key column

IDENTITY – Column is assigned unique values (identity in SQL7, uses a sequence in Oracle)

| Table: EDocuments | | | |
|--|----------|--------|---|
| Stores uploaded document content (currently unused). | | | |
| Column Name | Datatype | Length | Description |
| DocumentID | varchar | 40 | PK: The system, method, and apparatus of the invention document ID. |
| DocumentBlob | image | | The document content. |

| Table: EPoll | | | |
|----------------------------|----------|--------|--|
| Stores responses to Polls. | | | |
| Column Name | Datatype | Length | Description |
| PollID | varchar | 40 | The system, method, and apparatus of the invention ObjectID of the Poll object. |
| UserID | varchar | 40 | The system, method, and apparatus of the invention ObjectID of the User who responded. |
| QuestionID | int | | The zero-based index of the question. |
| Response | varchar | 100 | Text response (reserved for future use). |
| Response1 | int | | Response to the first choice (one if selected, zero if not). |
| Response2 | int | | Response to the second choice (one if selected, zero if not). |
| Response3 | int | | Response to the third choice (one |

| | | | |
|---|-----|--|--|
| | | | if selected, zero if not). |
| Response4 | int | | Response to the fourth choice (one if selected, zero if not). |
| Response5 | int | | Response to the fifth choice (one if selected, zero if not). |
| Response6 | int | | Response to the sixth choice (one if selected, zero if not). |
| Response7 | int | | Response to the seventh choice (one if selected, zero if not). |
| Response8 | int | | Response to the eighth choice (one if selected, zero if not). |
| Response9 | int | | Response to the ninth choice (one if selected, zero if not). |
| Response10 | int | | Response to the tenth choice (one if selected, zero if not). |
| Index: EIndex (PollID, UserID, QuestionID) | | | |

| Table: EAudit Stores information about when data was accessed and by whom. | | | |
|--|----------|--------|--|
| Column Name | Datatype | Length | Description |
| AuditID | int | | PK, IDENTITY: The unique ID of this audit record. |
| ObjectID | varchar | 40 | Typically, the ObjectID of the object that was accessed. In some cases, such as failed login attempts, this will be the ID of the System object (3). |
| ObjectName | varchar | 512 | The name of the object corresponding to ObjectID. |

| | | | |
|------------------------------------|----------|-----|---|
| TargetID | varchar | 40 | The ObjectID of a secondary object that was involved in the audit event. |
| UserID | varchar | 40 | Typically, the UserID of the user that accessed the object. In some cases, such as failed login attempts, this will be the ID of the Admin user (11). |
| AuditAction | varchar | 32 | A short string representing the action that was audited, such as login, open, delete, etc. A list of these strings is in EAudit.java. |
| AuditDate | datetime | | The date and time of the audit event. |
| Detail | varchar | 255 | Additional detail text. If the request was initiated by an HTTP client, this is the HTTP result code (e.g. 200 for success). |
| ClientAddress | varchar | 255 | If the request was initiated by an HTTP client, this is the IP address of the client. |
| UserAgent | int | | If the request was initiated by an HTTP client, this is the UserAgent of the client (references EBrowserAgent.BrowserID). |
| Index: EAudit_Index1 (ObjectID) | | | |
| Index: EAudit_Index2 (UserID) | | | |
| Index: EAudit_Index3 (AuditDate) | | | |

Table: EBrowserAgent

Stores the names of browser user-agents.

| Column Name | Datatype | Length | Description |
|---|----------|--------|--|
| BrowserID | int | | IDENTITY, PK: The unique ID of this record. |
| BrowserAgent | varchar | 255 | The name of the HTTP client user-agent (as sent by the browser). |
| Unique Index: EBrowserAgent_Index1 (BrowserAgent) | | | |

Table: EPlayerAgent

Stores the names of Eloquent Player clients.

| Column Name | Datatype | Length | Description |
|---|----------|--------|--|
| PlayerID | int | | IDENTITY, PK: The unique ID of this record. |
| PlayerAgent | varchar | 255 | The name of the Player (as sent by the player plug-in or control). |
| Unique Index: EPlayerAgent_Index1 (PlayerAgent) | | | |

Table: EViewings

Stores information about talk viewings.

| Column Name | Datatype | Length | Description |
|-------------|----------|--------|---|
| AuditID | int | | A reference to the EAudit record that corresponds to this record. The EAudit record contains basic |

| | | | |
|----------------|----------|-----|--|
| | | | information about when and by whom the information was accessed. The record in this table contains additional Talk-specific data. |
| EloTalkID | varchar | 16 | The Eloquent Talk ID of the talk that was viewed. Do not confuse this with the system, method, and apparatus of the invention ObjectID of the talk, which can be found in EAudit.ObjectID. |
| PlayerAgentID | int | | The name of the Player agent that was used to view the talk (references EPlayerAgent.PlayerID). |
| OpenTime | datetime | | The date and time the talk was opened. |
| CloseTime | datetime | | The date and time the talk was closed. |
| ClientAddress | varchar | 255 | The IP address of the client that viewed the talk. |
| BytesSent | int | | The number of bytes streamed from the media server to the client. |
| Flags | int | | Bit mask that represents viewing attributes. |
| CompletionCode | int | | Zero = successful Non-zero = failed |
| TalkPercent | int | | The percentage of the talk that was viewed. |

1050390-991111

| | | | |
|---------------------------------------|-----|--|--|
| SlidesPercent | int | | The percentage of the slides that were viewed. |
| Duration | int | | The amount of time spent viewing. |
| ConnectTime | int | | The difference between OpenTime and CloseTime, expressed in number of seconds. |
| Kbps | int | | The bandwidth used by this viewing, as expressed in kilobits per second. |
| Unique Index: EViewings_Index1 (Kbps) | | | |

| Table: EQuizResponses | | | |
|--|----------|--------|--|
| Stores information about quiz response data. A record in this table may have many EQuizAnswers records associated with it. | | | |
| Column Name | Datatype | Length | Description |
| QuizResponseID | int | | IDENTITY, PK: The unique ID of this record. |
| AuditID | int | | A reference to the EAudit record that corresponds to this record. The EAudit record contains basic information about when and by whom the information was accessed. The record in this table contains additional Quiz-specific data. |
| EloTalkID | varchar | 16 | The Eloquent Talk ID of the talk that was viewed. Do not confuse this with the system, |

| | | | |
|----------------|----------|-----|---|
| | | | method, and apparatus of the invention ObjectID of the talk, which can be found in EAudit.ObjectID. |
| PlayerAgentID | int | | The name of the Player agent that was used to view the talk and take the quiz (references EPlayerAgent.PlayerID). |
| OpenTime | datetime | | The date and time the talk was opened. |
| CloseTime | datetime | | The date and time the talk was closed. |
| ClientAddress | varchar | 255 | The IP address of the client that viewed the talk and took the quiz. |
| BytesSent | int | | The number of bytes streamed from the media server to the client. |
| Flags | int | | Bit mask that represents viewing attributes. |
| CompletionCode | int | | Zero = successful Non-zero = failed |
| NumQuestions | int | | The number of total questions in the quiz. |
| NumWithAnswers | int | | The number of questions in the quiz that define a correct answer. |
| Grade | int | | The grade for this quiz response, expressed as a percentage, and defined as the |

| | | | |
|--|--|--|---|
| | | | number of correct answers divided by the number of questions in the quiz. |
|--|--|--|---|

| Table: EQuizAnswers | | | |
|---|----------|--------|---|
| Each record represents one answer to one quiz question. | | | |
| Column Name | Datatype | Length | Description |
| QuizResponseID | int | | PK[1]: The ID of the QuizResponse that this answer is associated with (references EQuizResponses.QuizResponseID). |
| AnswerNumber | smallint | | PK[3]: The zero-based answer number for this particular QuizResponseID and QuestionID. |
| QuestionID | smallint | | PK[2]: The number of the question that is answered by this record. |
| QuestionType | smallint | | The type of question: 0 = multiple choice 1 = essay |
| Correct | smallint | | 0 = incorrect 1 = correct 2 = not applicable |
| MultiChoiceAnswer | smallint | | The multiple choice answer that was selected by the user. |
| EssayAnswer | varchar | 2000 | The essay answer. |

Views

Views are used in the system, method, and apparatus of the invention to isolate database-specific constructs and to aggregate information for reporting.

| View: EUsageView | | | |
|---|----------|--------|--|
| A view that joins EAudit, EViewings, and EIndex to consolidate information for usage reports. | | | |
| Column Name | Datatype | Length | Description |
| AuditID | int | | The ID of the audit record (references EAudit.AuditID). |
| ObjectID | varchar | 40 | The system, method, and apparatus of the invention Object ID of the Talk that was viewed. |
| UserID | varchar | 40 | The system, method, and apparatus of the invention Object ID of the User that viewed the talk. |
| UserName | varchar | 256 | The Name corresponding to UserID (references EIndex.FieldValue where EIndex.FieldName='Name') |
| EloTalkID | varchar | 16 | The Eloquent Talk ID (references EViewings.EloTalkID). |
| TalkName | varchar | 256 | The Name corresponding to Object ID (references EIndex.FieldValue where EIndex.FieldName='Name'). |
| EventID | varchar | 40 | The system, method, and apparatus of the invention Object ID of the Event that contains the talk (references EIndex.FieldValue |

| | | | |
|---------------|----------|-----|--|
| | | | where EIndex.FieldName='ParentID') |
| EventName | varchar | 256 | The Name corresponding to EventID (references EIndex.FieldValue where EIndex.FieldName='SourceEvent') |
| OpenTime | datetime | | The date and time the talk was opened. |
| CloseTime | datetime | | The date and time the talk was closed. |
| ConnectTime | int | | The difference between OpenTime and CloseTime, expressed in number of seconds. |
| Kbps | int | | The bandwidth used by this viewing, as expressed in kilobits per second. |
| ClientAddress | varchar | 255 | The IP address of the client that viewed the talk. |
| BytesSent | int | | The number of bytes streamed from the media server to the client. |
| TalkPercent | int | | The percentage of the talk that was viewed. |
| SlidesPercent | int | | The percentage of the slides that were viewed. |
| Duration | int | | The amount of time spent viewing. |

View: EQuizResponseView

A view that joins EAudit, EQuizResponses, EQuizAnswers, and EIndex to consolidate information for quiz reports. Due to limitations in some database systems, this view is implemented using two subsidiary views.

EQuizResponseView1 and EQuizResponseView2.

| Column Name | Datatype | Length | Description |
|-------------|----------|--------|--|
| AuditID | Int | | The ID of the audit record (references EAudit.AuditID). |
| ObjectID | Varchar | 40 | The system, method, and apparatus of the invention Object ID of the Talk that was viewed. |
| UserID | Varchar | 40 | The system, method, and apparatus of the invention Object ID of the User that viewed the talk. |
| UserName | Varchar | 256 | The Name corresponding to UserID (references EIndex.FieldValue where EIndex.FieldName='Name') |
| EloTalkID | Varchar | 16 | The Eloquent Talk ID (references EViewings.EloTalkID). |
| TalkName | Varchar | 256 | The Name corresponding to Object ID (references EIndex.FieldValue where EIndex.FieldName='Name'). |
| EventID | varchar | 40 | The system, method, and apparatus of the invention Object ID of the Event that contains the talk (references EIndex.FieldValue where |

| | | | |
|----------------|----------|-----|---|
| | | | EIndex.FieldName='ParentID') |
| EventName | Varchar | 256 | The Name corresponding to EventID (references EIndex.FieldValue where EIndex.FieldName='SourceEvent') |
| OpenTime | Datetime | | The date and time the talk was opened. |
| CloseTime | Datetime | | The date and time the talk was closed. |
| ClientAddress | Varchar | 255 | The IP address of the client that viewed the talk. |
| NumQuestions | Int | | The number of total questions in the quiz. |
| NumWithAnswers | Int | | The number of questions in the quiz that define a correct answer. |
| Grade | Int | | A grade computed as the number of correctly answered questions divided by NumQuestions times a hundred. For example, if half the questions were answered correctly, Grade would have a value of 50. |

Stored Procedures and Functions

5 Stored procedures are used in the system, method, and apparatus of the invention to isolate database-specific constructs (such as SQL7 identity vs. Oracle sequence), or to group statements in order to minimize roundtrips between our code and the database system.

Procedure: insert_audit_record

Insert a new record into EAudit. If necessary, a new record will also be inserted into EBrowserAgent.

Return: the system-generated ID of the new record (EAudit.AuditID).

| Parameter | | | |
|---------------|----------|--------|---|
| Name | Datatype | Length | Description |
| objectID | Varchar | 40 | The system, method, and apparatus of the invention Object ID of the object being audited. |
| objectName | Varchar | 40 | The name of the object corresponding to objected. |
| targetID | Varchar | 40 | The ObjectID of a secondary object that was involved in the audit event. |
| userID | Varchar | 40 | The system, method, and apparatus of the invention Object ID of the user performing the action. |
| auditAction | Varchar | 32 | A short string representing the action that was audited, such as login, open, delete, etc. A list of these strings is in EAudit.java. |
| detail | Varchar | 255 | Additional detail information. |
| clientAddress | Varchar | 255 | The IP Address of the client. |
| userAgent | Varchar | 255 | The client's user-agent. |

Procedure: insert viewing record

Insert a new record into Viewings. If necessary, a new record will also be inserted into EPlayerAgent.

Return: none.

| Parameter | | | |
|----------------|----------|--------|---|
| Name | Datatype | Length | Description |
| auditID | Integer | | The EAudit record that is associated with this viewing (references EAudit.AuditID). |
| eloTalkID | Varchar | 16 | The Eloquent Talk ID. |
| playerAgent | Varchar | 255 | The Player's agent string. |
| openTime | Datetime | | The date and time the talk was opened. |
| closeTime | Datetime | | The date and time the talk was closed. |
| clientAddress | Varchar | 255 | The IP Address of the client. |
| bytesSent | Integer | | The number of bytes streamed from the media server to the client. |
| flags | Integer | | Bit mask that represents viewing attributes. |
| completionCode | Integer | | Zero = successful Non-zero = failed |
| talkPercent | Integer | | The percentage of the talk that was viewed. |
| slidesPercent | Integer | | The percentage of the slides that were viewed. |
| duration | integer | | The amount of time spent |

| | | | |
|-------------|---------|--|--|
| | | | viewing. |
| ConnectTime | Integer | | The difference between OpenTime and CloseTime, expressed in number of seconds. |
| Kbps | Integer | | The bandwidth used by this viewing, as expressed in kilobits per second. |

Procedure: insert_quiz_response_record

Insert a new record into EQuizResponses. If necessary, a new record will also be inserted into EPlayerAgent.

Return: the system-generated ID of the new record (EAudit.AuditID).

| Parameter | | | |
|---------------|----------|--------|---|
| Name | Datatype | Length | Description |
| auditID | integer | | The EAudit record that is associated with this quiz response (references EAudit.AuditID). |
| eloTalkID | varchar | 16 | The Eloquent Talk ID. |
| playerAgent | varchar | 255 | The Player's agent string. |
| openTime | datetime | | The date and time the talk was opened. |
| closeTime | datetime | | The date and time the talk was closed. |
| clientAddress | varchar | 255 | The IP Address of the client. |
| bytesSent | integer | | The number of bytes streamed from the media server to the client. |

| | | | |
|----------------|---------|--|---|
| flags | integer | | Bit mask that represents viewing attributes. |
| completionCode | integer | | Zero = successful Non-zero = failed |
| numQuestions | integer | | The number of total questions in the quiz. |
| numWithAnswers | integer | | The number of questions in the quiz that define a correct answer. |
| grade | integer | | The grade for this quiz response. |

Procedure: insert_new_audit_record

Insert a new record into EAudit. This procedure differs from insert_audit_record in that it allows the programmer to specify an audit date, rather than using the current server date and time, and it requires the user-agent to be specified as an index into EBrowserAgent, rather than as a string. This procedure is used only when importing audit records from an ECP 5.0 installation.

Return: the system-generated ID of the new record (EAudit.AuditID).

| Parameter Name | Datatype | Length | Description |
|----------------|----------|--------|---|
| objectID | vchar | 40 | The system, method, and apparatus of the invention Object ID of the object being audited. |
| objectName | vchar | 40 | The name of the object corresponding to objectID. |
| targetID | vchar | 40 | The ObjectID of a secondary |

| | | | |
|---------------|----------|-----|---|
| | | | object that was involved in the audit event. |
| userID | varchar | 40 | The system, method, and apparatus of the invention Object ID of the user performing the action. |
| auditAction | varchar | 32 | A short string representing the action that was audited, such as login, open, delete, etc. A list of these strings is in EAudit.java. |
| auditDate | DateTime | | The date the audit event occurred. |
| detail | varchar | 255 | Additional detail information. |
| clientAddress | varchar | 255 | The IP Address of the client. |
| userAgent | integer | | The client's user-agent of the original audit event, specified as a reference to EBrowserAgent.BrowserID. |

Procedure: insert_browser_agent_record

Insert a new record into EBrowserAgent. This procedure is used only when importing audit records from an ECP 5.0 installation.

Return: the system-generated ID of the new record
(EBrowserAgent.BrowserID).

| Parameter | | | |
|--------------|----------|--------|-----------------------------|
| Name | Datatype | Length | Description |
| browserAgent | varchar | 255 | The HTTP user agent string. |

Procedure: insert_player_agent_record

Insert a new record into EPlayerAgent. This procedure is used only when importing audit records from an ECP 5.0 installation.

Return: the system-generated ID of the new record (EPlayerAgent.PlayerID).

| Parameter Name | Datatype | Length | Description |
|-------------------|----------|--------|-----------------------------------|
| browserAgent | varchar | 255 | The Eloquent player agent string. |

5 The Table "Objects" stores object definitions as a blob of XML data, including the ObjectID of the object, the version of the object, and the XML object definition. The table Eindex stores selected attributes for each object, including the ObjectID of the object, the attribute nname, and the attribute value. Eobjects is a persistent object store that contains entire object definition, while Eindex is an index of object attributes. Updating the Eindex table does not change the object definition. The Eindex table can be
10 used to locate objects (including parent/child relationships). Examples of Eobjects include:

select ObjectBlob from EObjects where ObjectID='1'

15 <Library>
 <ID>1</ID>
 <Name>ECS Library</Name>
 <Permissions_ SIZE="2">
 <Permissions KEY="5">1</Permissions>
 20 <Permissions KEY="0">1</Permissions>
 </Permissions_>

5 <DateCreated>2000/06/28 20:12:28 GMT</DateCreated>
 <DateModified>2000/06/28 20:12:28 GMT</DateModified>
 <WhoModified>11</WhoModified>
 <WhoCreated>11</WhoCreated>
 <Version>1</Version>
 <MimeType>object/library</MimeType>
 <TimepointEnabled>>false</TimepointEnabled>
 <Author></Author>
 <Organization></Organization>
 10 <Abstract></Abstract>
 <GeneratedAbstract></GeneratedAbstract>
 </Library>

while an example of Eobjects is

15 select FieldName, FieldValue from EIndex where ObjectID='1'

| | FieldName | FieldValue |
|----|--------------|-------------------------|
| | ----- | |
| 20 | AttachedID | [null] |
| | Author | |
| | ContentDate | [null] |
| | DateCreated | 2000/06/28 20:12:28 GMT |
| | DateModified | 2000/06/28 20:12:28 GMT |
| 25 | ID | 1 |
| | LastModified | [null] |
| | MimeType | object/library |
| | Name | ECS Library |
| | Organization | |
| 30 | ParentID | [null] |
| | Permissions | 0 |

| | |
|-------------|---------|
| Permissions | 5 |
| Type | Library |
| WhoCreated | 11 |
| WhoModified | 11 |

5

The table: Esystem stores database and system properties, such as system keys and system values.

10 The table EAudit stores a record of system activity, such as AuditID, Object ID, ObjectName, TargetID, and UserID, where Audit ID is the unique ID of an audit record, ObjectID is the ID of the Object that was accessed, ObjectName is the name of the object corresponding to the ObjectID, the TargetID is the ObjectID of a secondary object involved in the audit event, and UserID is the ID of the User that accessed the object. AuditAction is a short string representing the action that was audited, such as a login, open, delete, etc. AuditDate is the date and time of the audit event. Detail is a text block for additional detail. ClientAddress is the IP address of the client, if any, initiating the request, while UserAgent is the UserAgent of the client initiating the request.

20 The Table EbrowserAgent stores the name of browser user-agents, including BrowserID, the unique ID of the record, and BrowserAgent, the name of the HTTP user-agent. Likewise, the table EplayerAgent stores the names of player clients, including PlayerID and PlayerAgent, the player's unique ID and name.

25 The Table: Eviewings Stores information about talk viewings. including the AuditID (a reference to the Eaudit record that corresponds to the talk, EloTalkID is the ID of the talk that was viewed, and PlayerAgentID is the name of the PlayerAgent that was used to view the talk. OpenTime and CloseTime are the times the talk was opened and closed. ClientAddress is the IP address of the client that viewed the talk, BytesSent is the number of bytes streamed from the media server to the client, Flags is a bit mask that represents viewing attributed.

30

Views aggregate information for reporting and isolate database-specific constructs, such as EusageView (for Eaudit, Eviewings, and Eindex) and EquizResponseView (Eaudit, EquizResponses, EquizAnswers, and Eindex).

5 The method, system, and apparatus of our invention is further characterized by the use of stored procedures. Stored procedures provide more efficient processing, isolate database-specific constructs (especially IDENTITY vs. Sequence), and allow for such functions as insert_audit_record, insert_viewing_record, and insert_quiz_response_record.

10

 The method, system, and apparatus of our invention utilizes various off the shelf and commercially available database tools (SQL7), such as Microsoft SQL Server 7, Enterprise Manager (to examine schema, examine data, and backup and restore), as well as Query Analyzer execute SQL statements, and Profiler to trace all database activity.

15

 Oracle database tools are also used, such as DBA Studio to examine schema, EXP, and IMP, as command line utilities for backup, restore, and SQL Worksheet and SQL Plus to execute SQL statements.

20

 Sybase Database Tools are also used, such as Sybase Central to provide access to other tools and utilities, and Interactive SQL to execute SQL statements and start a database server.